CS 240

Data Structures and Algorithms I

Alex Vondrak

ajvondrak@csupomona.edu

October 14, 2011

Converting Infix To Postfix

- If you see a left parenthesis, push it onto the stack
- If you see a number, write it to the output
- If you see an operator, push it onto the stack
- Otherwise, next symbol should be a right parenthesis, and the top of the stack should be an operator
    - Pop the operator and write it to the output
    - Top of the stack should be a left parenthesis, so pop and discard
- At the end of the input, stack should be empty

Examples

- `((1 + 2) * 3)`
- `((1 + 2) * (3 + 4))`

Converting Infix To Postfix

- If you see a left parenthesis, push it onto the stack
- If you see a number, write it to the output
- If you see an operator, push it onto the stack
- Otherwise, next symbol should be a right parenthesis, and the top of the stack should be an operator
  - Pop the operator and write it to the output
  - Top of the stack should be a left parenthesis, so pop and discard
- At the end of the input, stack should be empty

Examples

- `((1 + 2 * 3)`
- `(1 + 2) * (3 + 4))`

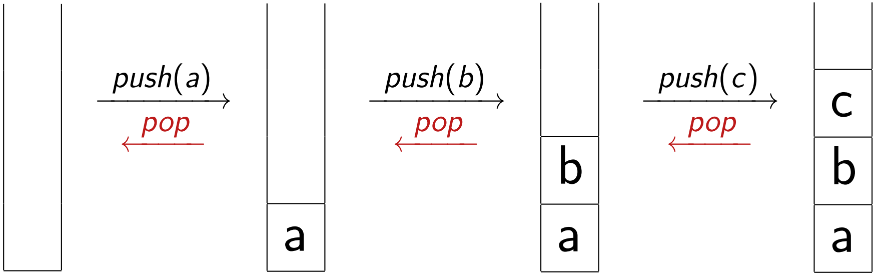
Stacks

Abstract Data Type

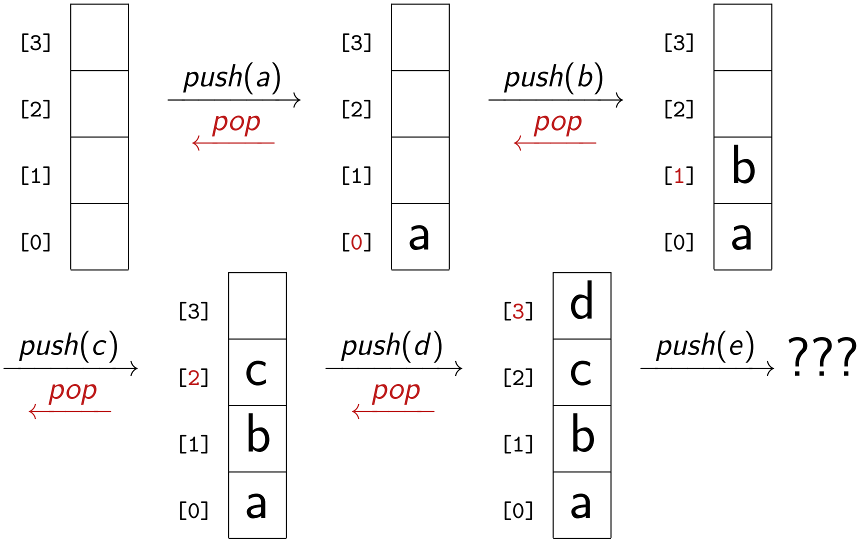
```
interface Stack {
    public void push(int item);
    public int pop()
        throws StackUnderflowException;
    public int top()
        throws StackUnderflowException;
    public boolean isEmpty();
    public int size();
}

class SomeStackImplementation implements Stack {
    /* must implement all the methods */
}
```

Stack Implementation



Stack Implementation



ArrayStack

```
class ArrayStack implements Stack {
   public void push(int item) { ... }

   public int pop()
      throws StackUnderflowException { ... }

   public int top()
      throws StackUnderflowException { ... }

   public boolean isEmpty() { ... }

   public int size() { ... }
}
```

ArrayStack

Constructor

```
class ArrayStack implements Stack {
   private int[] data;
   private int top;

   public ArrayStack() {
      final int CAPACITY = 10;
      top = -1;
      data = new int[CAPACITY];
   }

   // ...
}
```

ArrayStack
Auxiliary Methods

```
class ArrayStack implements Stack {
  // ...

  public int size() {
     return top + 1;
  }


  public boolean isEmpty() {
     return (size() == 0);
  }

  // ...
}
```

ArrayStack
top()

```
class ArrayStack implements Stack {
    // ...

    public int top() throws StackUnderflowException
        if (isEmpty())
            throw new StackUnderflowException();
        return data[top];
    }

    // ...
}
```

ArrayStack
pop()

```
class ArrayStack implements Stack {
    // ...

    public int pop() throws StackUnderflowException
        if (isEmpty())
            throw new StackUnderflowException();

        int result = top();
        top--;
        return result;
    }

    // ...
}
```